PYRAMID: An Object-Based Library for Parallel Unstructured Adaptive Mesh Refinement

Charles D. Norton, John Z. Lou, and Thomas A. Cwik

National Aeronautics and Space Administration

Jet Propulsion Laboratory

California Institute of Technology

http://hpc.jpl.nasa.gov/APPS/AMR

HPCC/Earth and Space Sciences Project

PYRAMID: Parallel Unstructured Adaptive Mesh Refinement

PYRAMID

Modern... Simple... Efficient... Scalable...

Technology Description

An advanced software library supporting parallel adaptive mesh refinement in large-scale, adaptive scientific & engineering simulations.

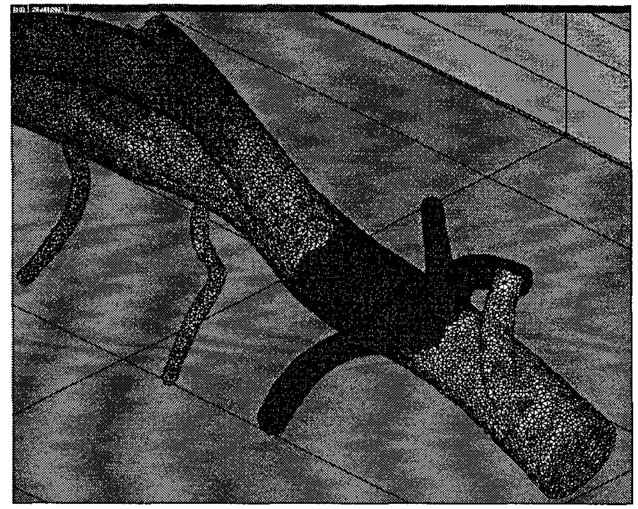
State-of-the-Art Design!

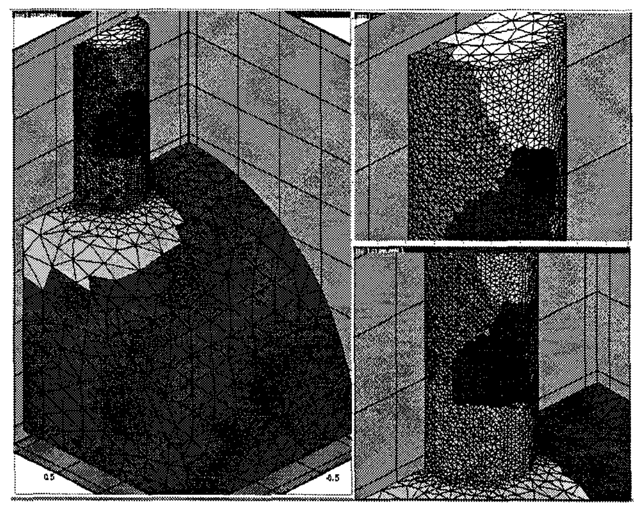
- Efficient object-oriented design in Fortran 90 and MPI
- Automatic mesh quality control & dynamic load balancing
- Scalable to hundreds of processors & millions of elements

Application Arena

- Computer Modeling & Simulation Applications with complex geometry
- Electromagnetic and semiconductor device modeling
- Structural/Mechanical/Fluid dynamics applications

John Z. Lou, Charles D. Norton, & Thomas A. Cwik
High Performance Computing Systems and Applications Group
http://hpc.jpl.nasa.gov/APPS/AMR



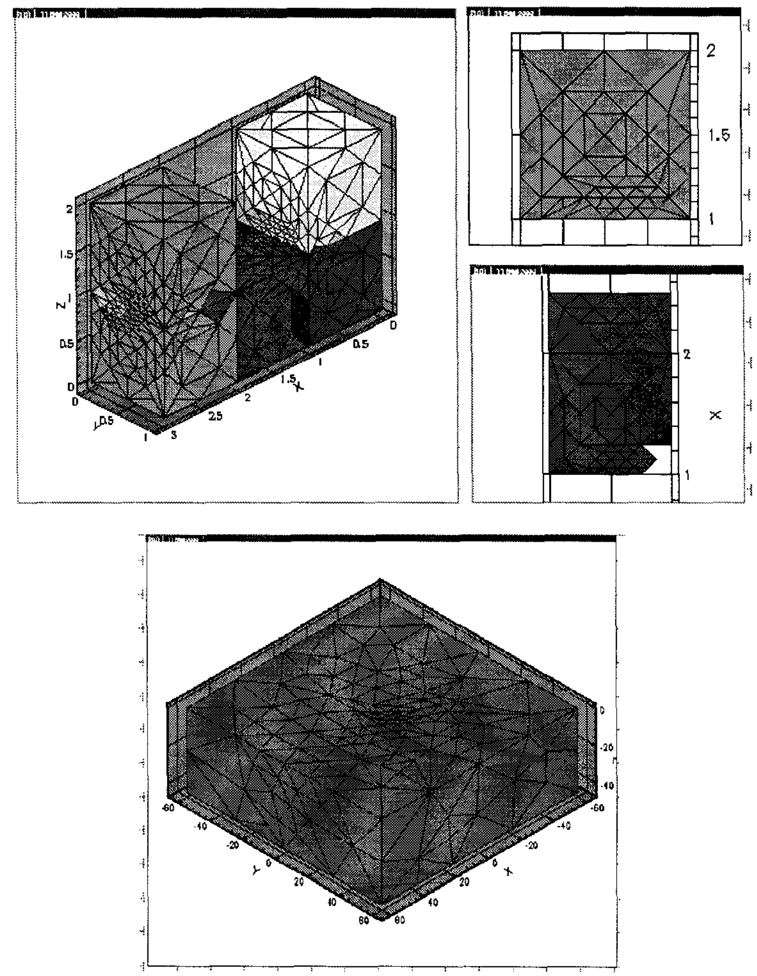


Initial geometry courtesy of SCOREC (Rensselaer)

Pyramid Package Components

- **Components**
  - Parallel mesh I/O, partitioner, logical and physical adaptive refiners, mesh migration, and visualization

- **Development Structure**
  - Fortran 90: Core data structures and internals
  - C: Interface to ParMetis graph partitioner
  - MPI: Distributed-Memory communication

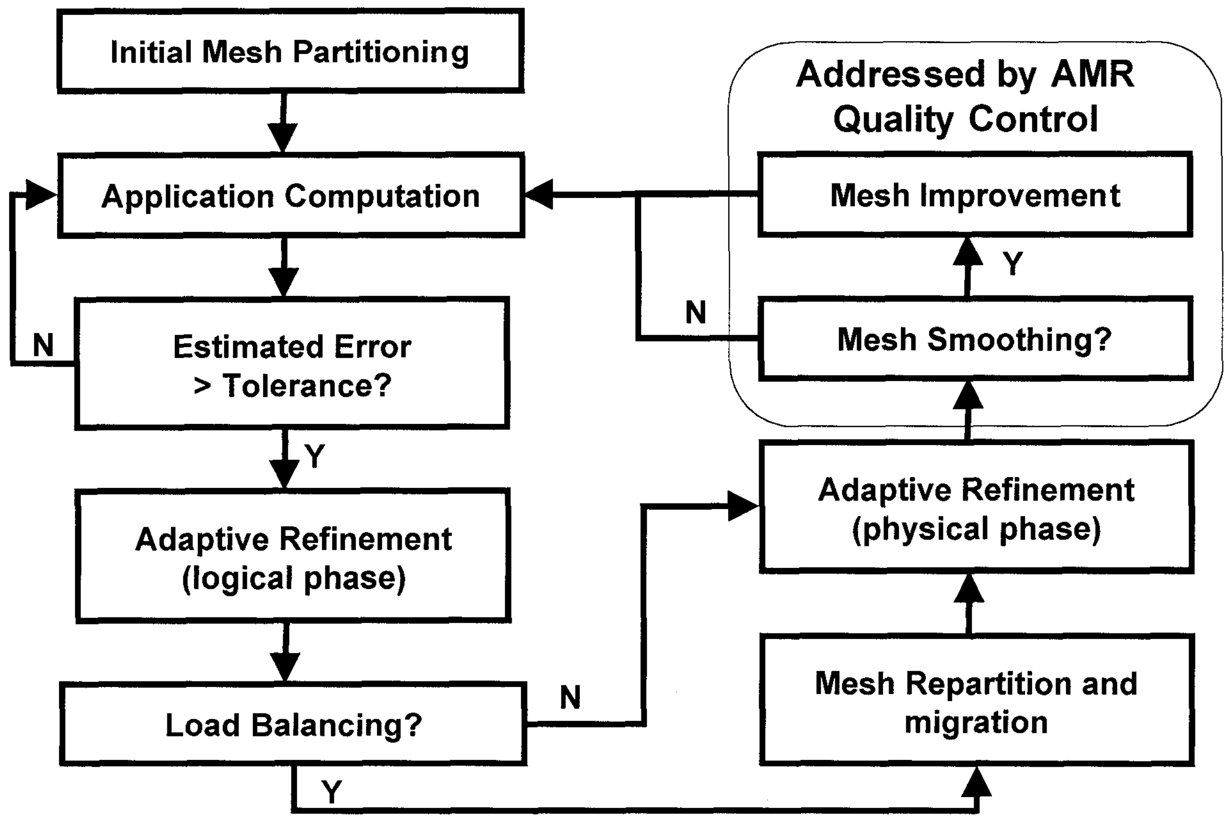


Development Issues

- **Parallel Unstructured AMR Scheme**
  - Logical AMR: Iterative scheme defining refinement pattern (with mesh quality control)
  - Physical AMR: Locally refine coarse elements based on logical refinement
- **Parallel Dynamic Load Balancing Strategy**
  - Repartition weighted logical mesh, migrate coarse elements, and perform local physical refinement
- **Modular Design**
  - Performance and abstraction features of Fortran 90

Our Parallel AMR Process

- **Organization**
  - Partitioning, Adaptive Refinement, Load Balancing, Mesh Migration, and Element Quality Control



Technology

- **Fortran 90/95 Features Modernize Programming**

Modules
Encapsulate data and routines across program units

Generic Interfaces
One call can perform different actions based on types

Array Syntax
Simplifies whole array, and array subset, operations

Use-Association
Controls access to module content

Derived Types
User-defined types supporting abstractions in programming

Pointers/Allocatable Arrays
Supports flexible/dynamic data structures

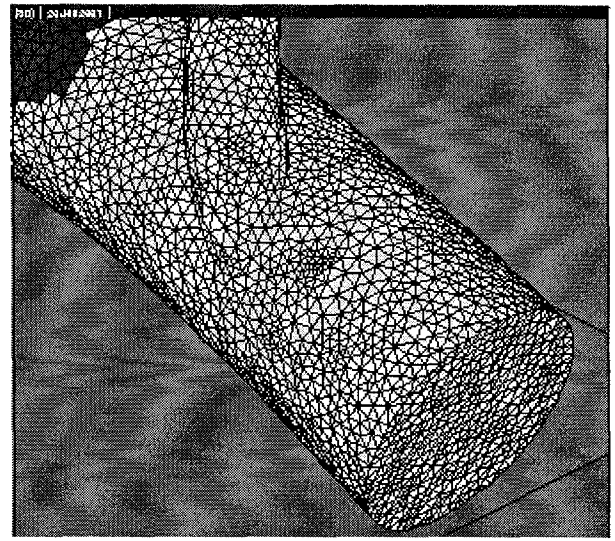
Backward compatible with Fortran 77

FOR MORE INFO...

Fortran 90 Programming. Ellis, Philips, & Lahey; Addison Wesley, 1994
http://hpc.jpl.nasa.gov/PEP/nortonc/oof90.html

Technology

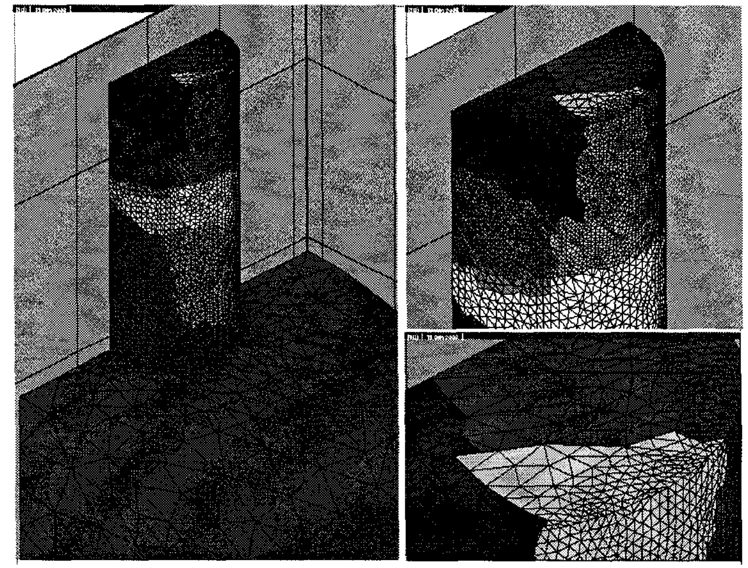
- **A Minimal PYRAMID Program**
  - Initialization Section
    - Optional arguments override defaults



```
PROGRAM pyramid_example
USE pyramid_module
implicit none
   ! Statements omitted...
   type (mesh), dimension(2) :: meshes
   call PAMR_INIT()
   call PAMR_LOAD_MESH_PARALLEL( meshes(1), in_file )
   call PAMR_REPARTITION( meshes(1) )
   ! Adaptive refinement loop…
   call PAMR_ELEMENT_COUNT( meshes(2) )
   call PAMR_VISUALIZE( meshes(2), "visfile.plt" )
   call PAMR_FINALIZE( mpi_active = .true. )
END PROGRAM pyramid_example
```

Technology

- **A Minimal PYRAMID Program**
  - Adaptive Refinement



```
PROGRAM pyramid_example
! Adaptive refinement loop...
   do i = 1, refinement_level
      call PAMR_ERROR_EST( meshes(1), &
                           meshes(2) )
      call PAMR_LOGICAL_AMR( meshes(1) )
      call PAMR_REPARTITION( meshes(1) )
      call PAMR_PHYSICAL_AMR( meshes(1), meshes(2) )
   end do
END PROGRAM pyramid_example
```

  - Users must specify their error estimation method
  - Mesh hierarchies can be defined

Technology

- **Object-Based Access to Data Structure**
  - Explicit reference to element coordinates is complicated

```
type (mesh) :: this
real, dimension(3) :: xyz_pos
xyz_pos = this%nodes(this%elements(2)%node_indx(1))%coord
```

  - PYRAMID simplifies such references

```
type (mesh) :: this
real, dimension(3) :: xyz_pos
real, dimension(3,4) :: all_pos
real, dimension(3,3,4) :: n_normal
xyz_pos  = PAMR_ELEMENT_COORD(this, element_indx=2, &
                                    node_indx=1)
all_pos  = PAMR_ELEMENT_COORD(this, element_indx=2)

! Access signed local normal basis for all faces
n_normal = PAMR_FACE_NORMALBASIS(this, element_indx=3)
```

Technology

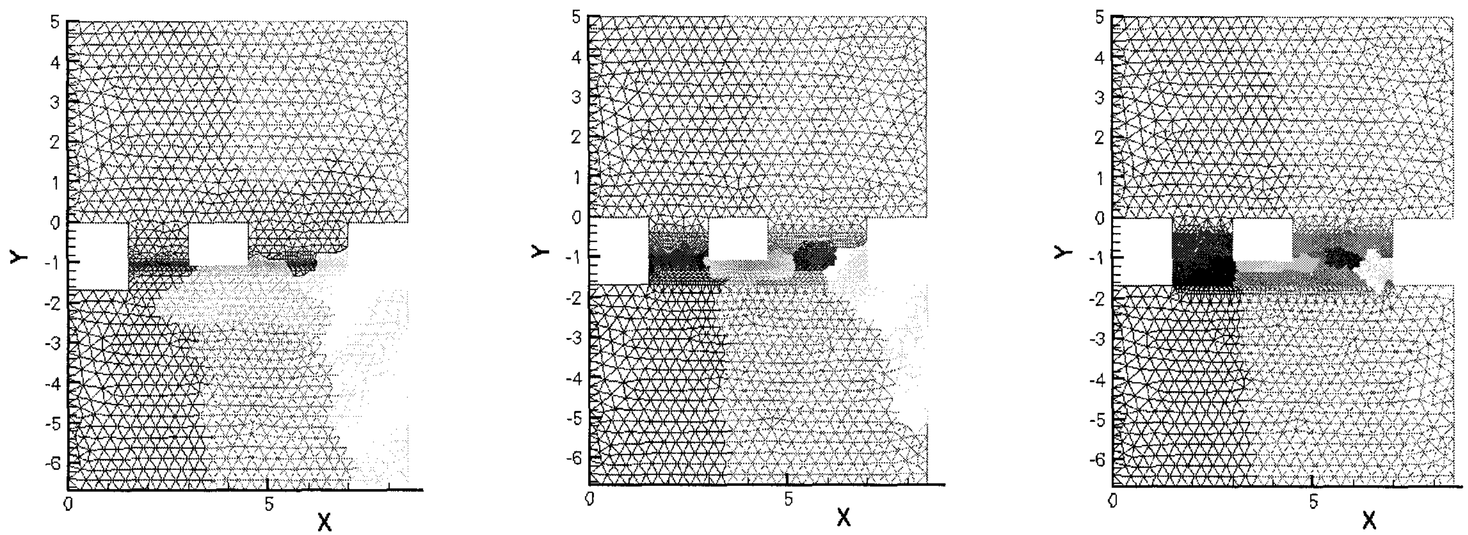
- **Numerous User-Driven Commands Are Included**
  - Initialization, Mesh I/O, Termination, Adaptive Refinement, Repartitioning, Data Migration, Visualization, Data Structure Access, Mathematical, and Auxiliary
  - Almost every command contains optional arguments for use customization

```
PAMR_CURRENT_TIME
PAMR_DEFINE_MESH_TERMS
PAMR_ELAPSED_TIME
PAMR_ELEMENT_CENTROID
PAMR_ELEMENT_COORD
PAMR_ELEMENT_COUNT
PAMR_ELEMENT_ID
PAMR_ELEMENT_VOLUME
PAMR_ERROR_EST
PAMR_FACE_AREA
PAMR_FACE_CENTROID
PAMR_FACE_COORD
PAMR_FACE_INDX
PAMR_FACEEDGE_ID
PAMR_FACEEDGE_INDX
PAMR_FACE_NORMALBASIS
PAMR_FACE_UNITNORMAL
PAMR_FINALIZE
PAMR_GET_EDGE_TERMS
PAMR_GET_ELEMENT_TERMS
PAMR_GET_FACE_TERMS
PAMR_GET_NODE_TERMS
PAMR_INIT
PAMR_LOAD_MESH_COMP
PAMR_LOAD_MESH_SERIAL
PAMR_LOAD_MESH_PARALLEL
PAMR_LOGICAL_AMR
PAMR_MAP_MESH_TERMS
PAMR_PHYSICAL_AMR
PAMR_REPARTITION
PAMR_SAVE_MESH
PAMR_SET_EDGE_TERMS
PAMR_SET_ELEMENT_TERMS
PAMR_SET_FACE_TERMS
PAMR_SET_NODE_TERMS
PAMR_VISUALIZE
```

  - Most commands are generic based on the mesh component applied

Technology

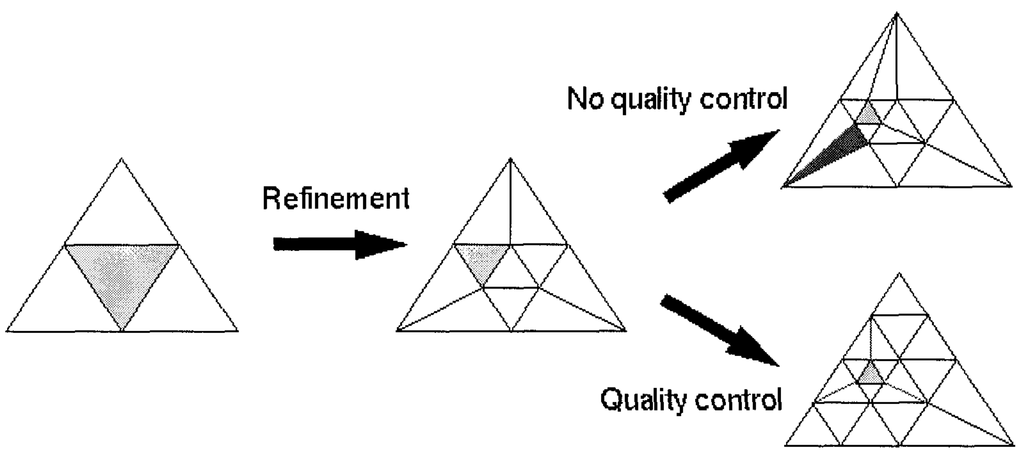
- **Dynamic Load Balancing with ParMetis**
  - ParMetis gives partitioning, PYRAMID performs migration
  - Migration handles irregular communication patterns with a scalable and efficient non-blocking algorithm

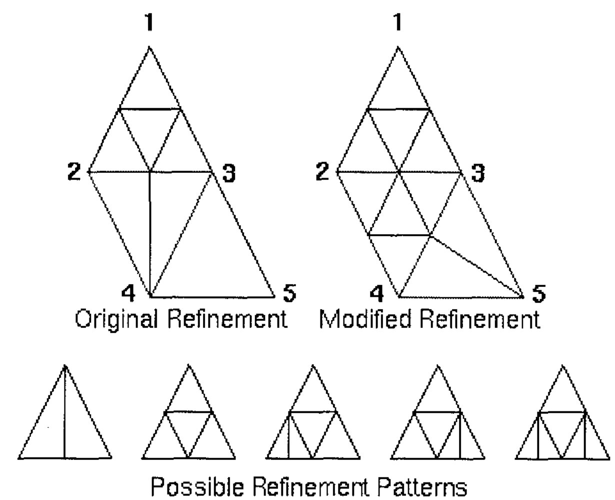


  - We are investigating Zoltan (Sandia National Labs) as an additional option for partitioning

Technology

- **Automatic Mesh Quality Control**
  - Modify coarse element refinement if successive refinements cause poor aspect ratios

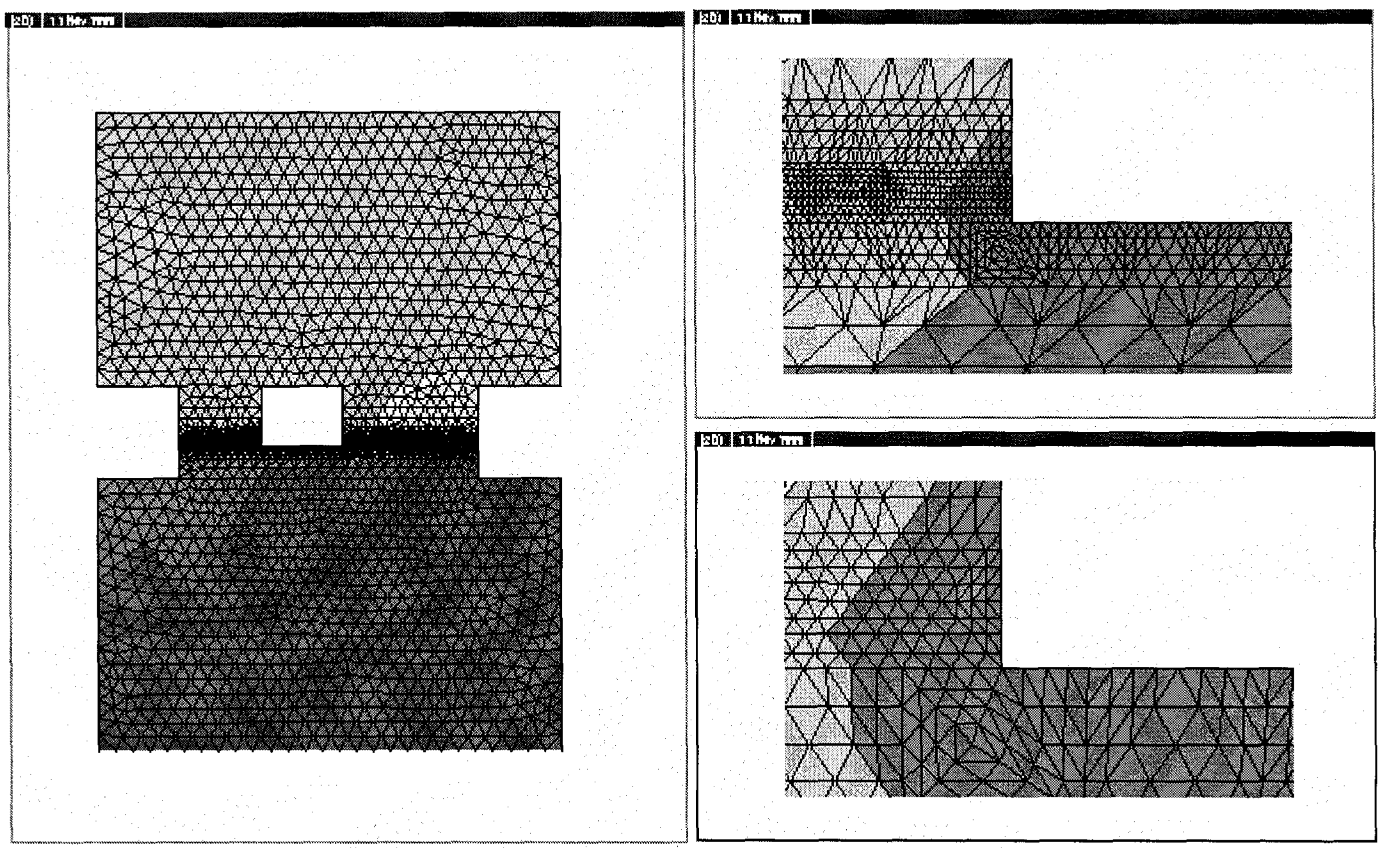




  - Controls quality at the expense of additional elements

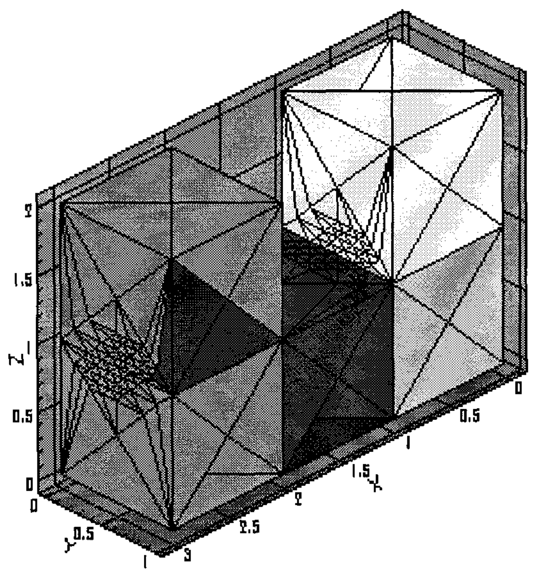
Technology

- **Automatic Mesh Quality Control**
  - Benefit of quality control applied to triangular elements

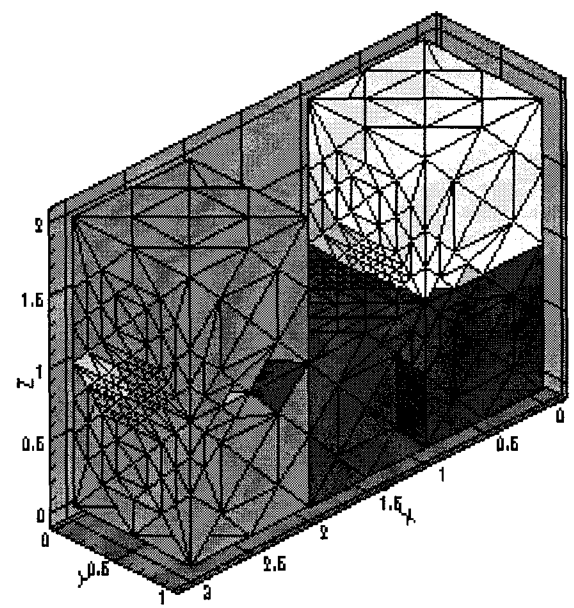


Technology

- **Automatic Mesh Quality Control**
  - Benefit of quality control applied to tetrahedral elements



Poor Mesh Elements Without
Quality Control

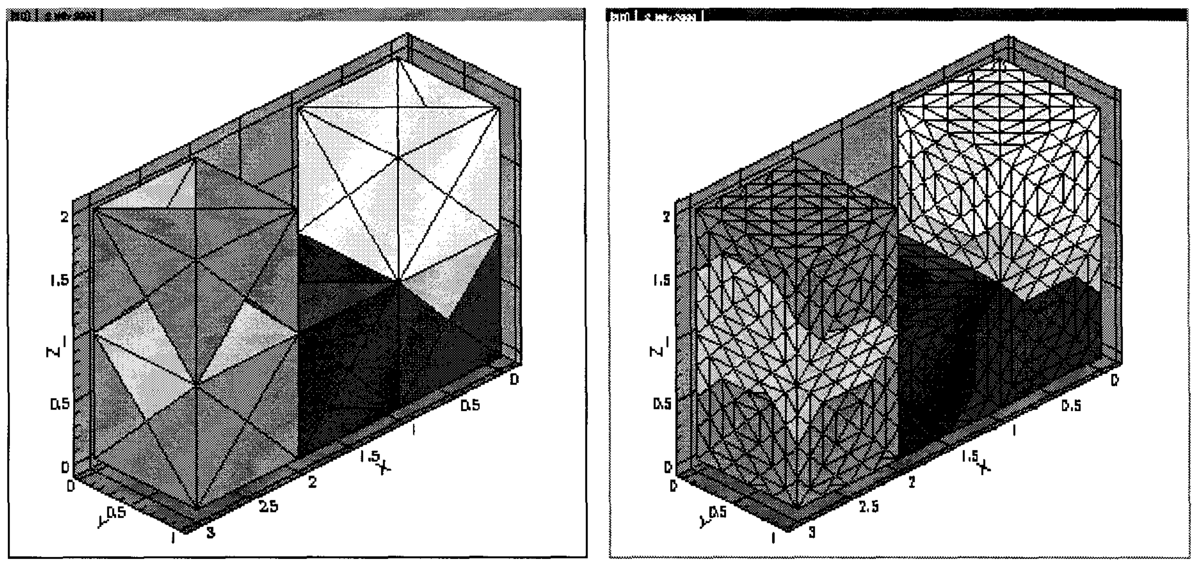


Good Mesh Elements With
Quality Control

Note : Tecplot shows some edges in the backplane that do not exist in the mesh...

Technology

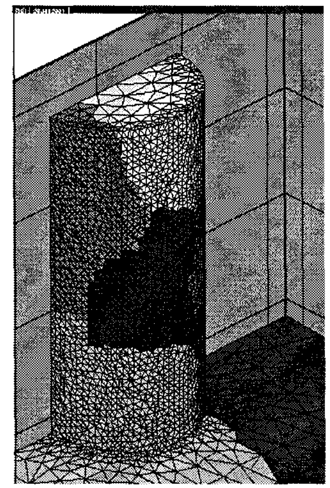
- **Large Scale Parallel Mesh Generation**
  - Specify uniform error for generation from coarse meshes



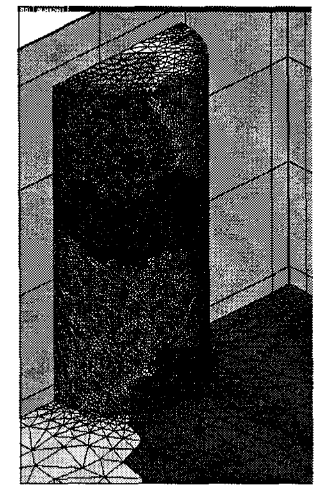
Parallel Uniform Refinement

Performance

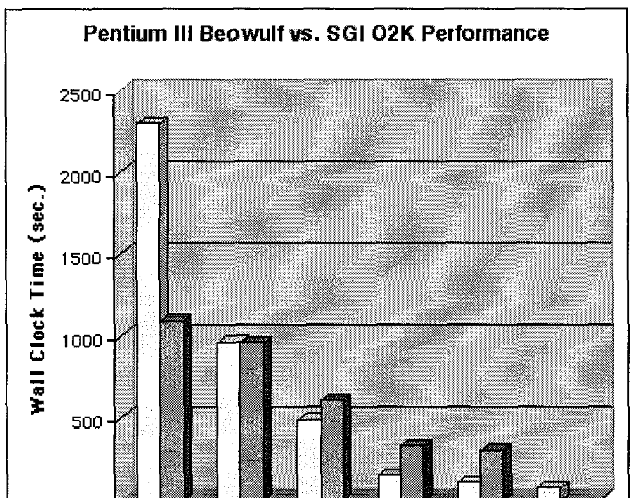
- **Pentium III Beowulf Cluster vs. SGI O2K Parallel AMR**
  - O2K scales well although the processor is slower than the 800 Mhz Beowulf PIII
  - Beowulf competes well, but performance is limited by 100 BaseT network



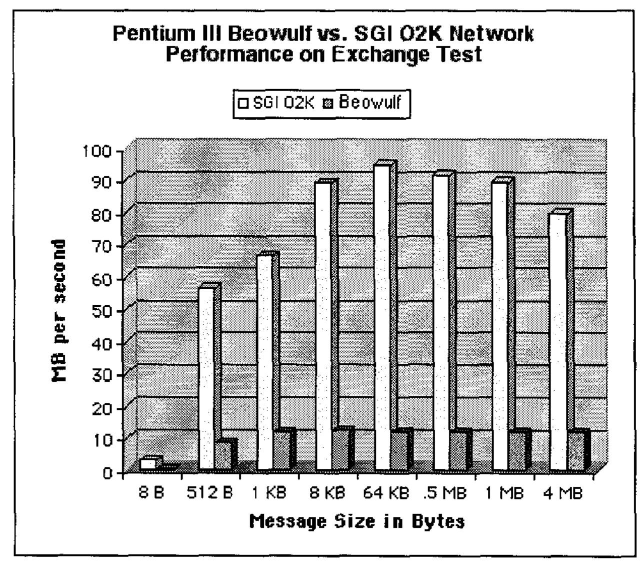
Repartitioned Muzzle Brake



After 3 Adaptive Refinements

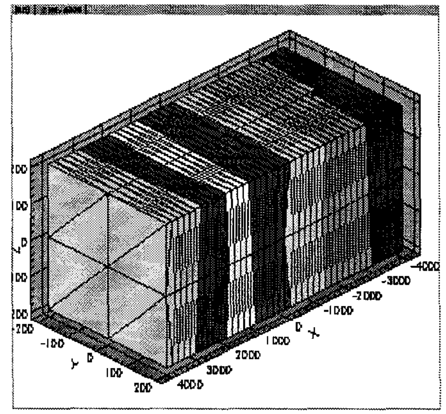


	4 Pes	8 Pes	16 Pes	32 Pes	40 Pes	64 Pes
SGI O2K	2295.5	964.68	486.32	148.42	106.5	77.01
Beowulf	1091.19	960.71	607.56	329.58	303.9	

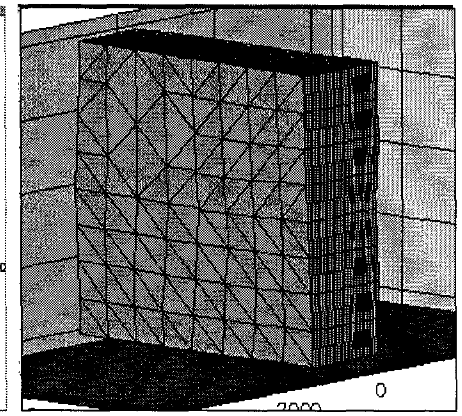


Performance

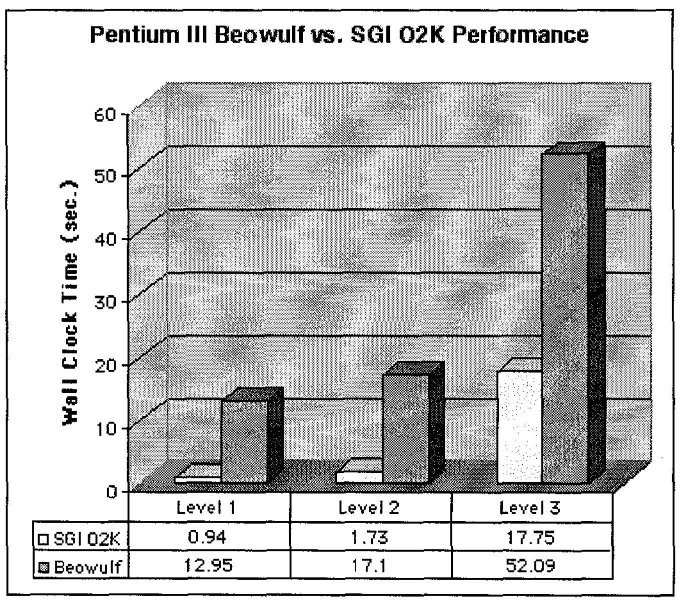
- **Pentium III Beowulf Cluster vs. SGI O2K Parallel AMR**
  - O2K outperforms Beowulf across all refinement levels
  - Beowulf shows larger percentage improvement as problem size grows



Earthquake Region Mesh



Refined Cross-Section

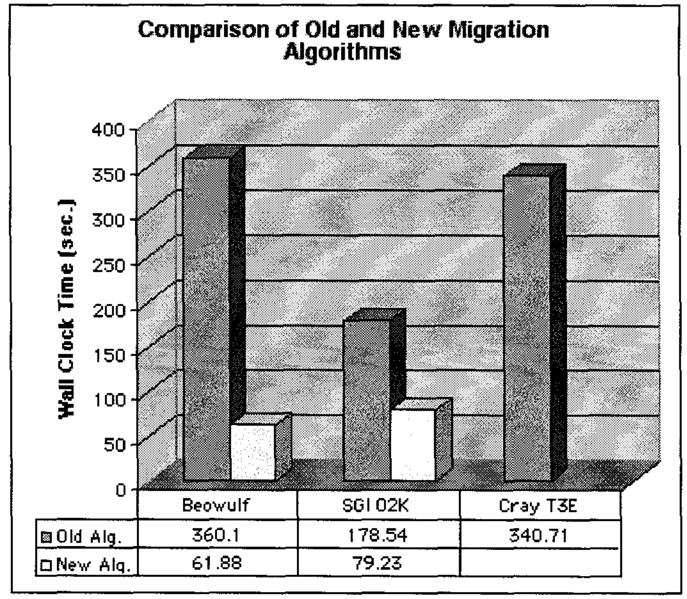


  - O2K is an order of magnitude slower from level 2 to level 3
  - Performance will vary based on mesh geometry

Note : Simulation uses 32 processors
New migration algorithms applied

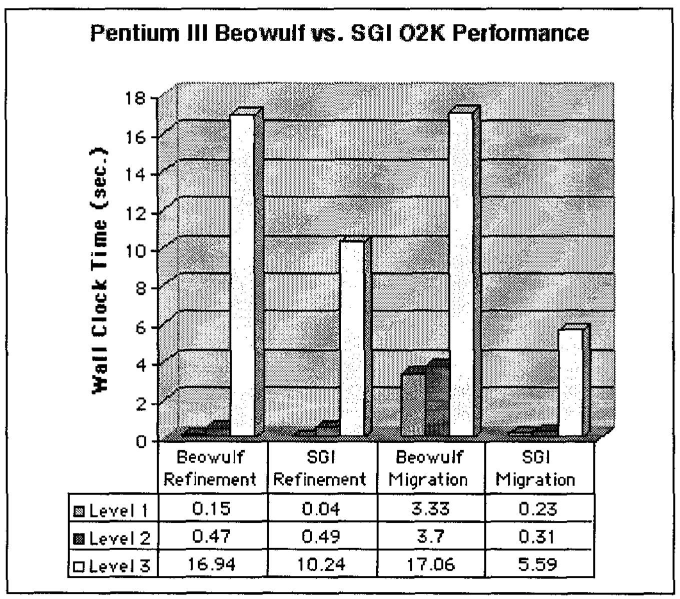
Performance

- **Pentium III Beowulf Cluster vs. SGI O2K Parallel AMR**
  - Migration algorithm improvements benefit Beowulf significantly
  - Network still hinders Beowulf with increased problem size



	Beowulf	SGI O2K	Cray T3E
Old Alg.	360.1	178.54	340.71
New Alg.	61.88	79.23	

Adaptively Refined Earthquake Mesh on 8 PEs
T3E decommissioned prior to this simulation



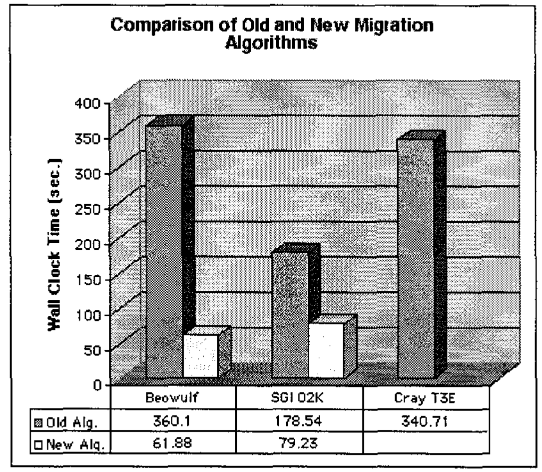
	Beowulf Refinement	SGI Refinement	Beowulf Migration	SGI Migration
Level 1	0.15	0.04	3.33	0.23
Level 2	0.47	0.49	3.7	0.31
Level 3	16.94	10.24	17.06	5.59

Earthquake Mesh Refinement and Migration on 32 PEs

- Our new migration algorithms are completely non-blocking, scalable, and utilize full-duplex channels when available
- We estimate O2K has a 7 times network speed advantage over 100 BaseT Beowulf

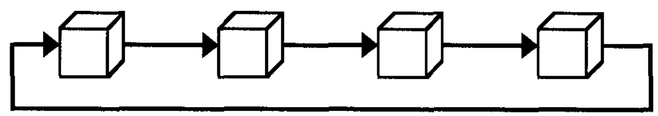
Performance

- **Irregular Data Communication**
  - Migration requires irregular, but predictable, data movement that varies in size and destination



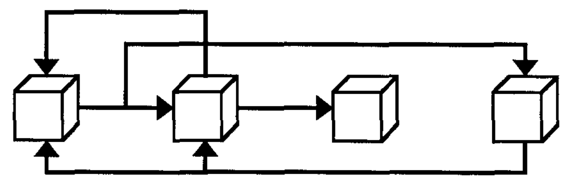
Circular-Shift "MPI_SENDRECV(...)"

- All processors inspect all of the data
- "Guarantees" handling of cyclic deadlock dependencies
- Irregular data sizes affect pipelined flow performance
- MPI implicit buffering, due to poor pipeline structure, leads to poor performance



Direct Data Transfers

- Processors send/receive specific messages
- Send continuously while checking for receives
- "Arbitrary" message ordering can flood the network switch, leading to poor performance



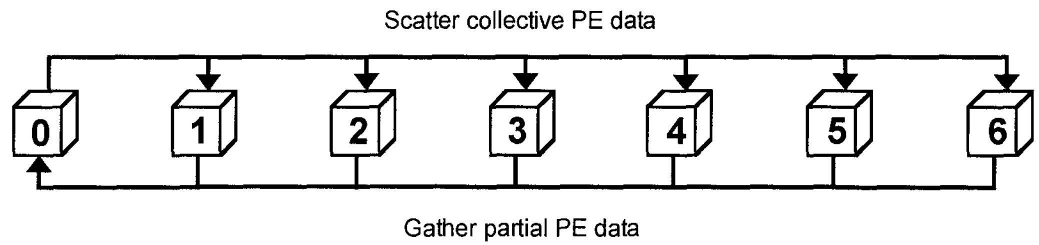
Performance

- **Irregular Data Communication**
  - Reduction schemes can improve performance, if implemented with care...

Note : Broadcasts simplify handling cases where the number of processors is not a power of two

MPI Reduce to 0 with Broadcast Scheme

- Not scalable and very inefficient for large data sets

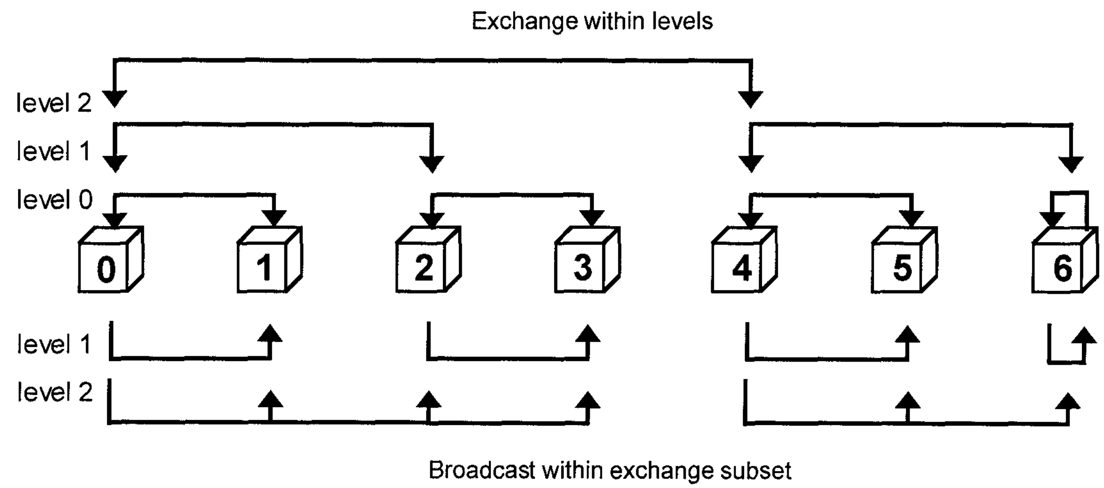


MPI Reduce to Leader with Subset Broadcast Scheme

- More scalable and efficient, but still requires multiple broadcasts at each tree level

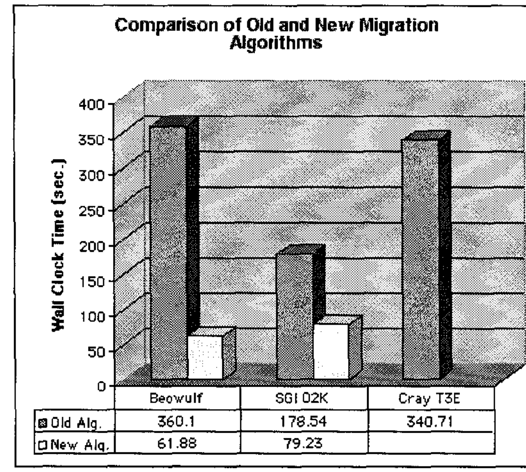
Exchange with Subset Broadcast

- Same characteristics as above



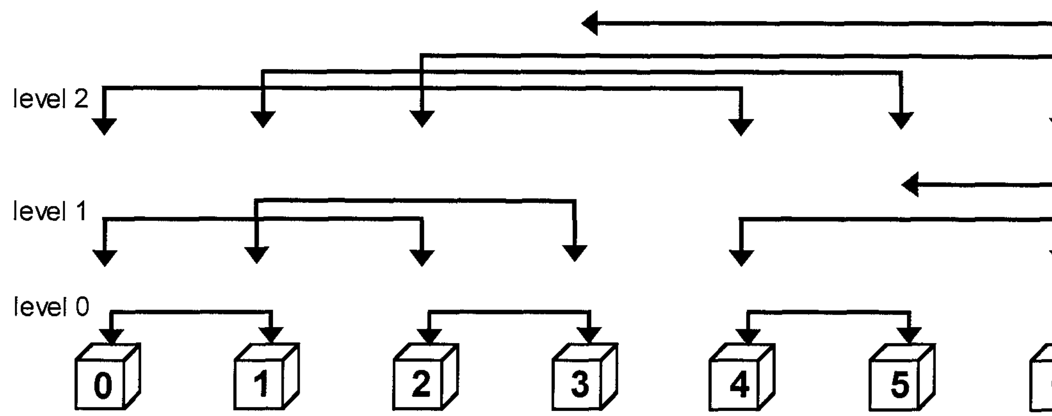
Performance

- **Irregular Data Communication**
  - Reduction schemes can improve performance, if implemented with care...



<u>Our Algorithm Improvements</u>

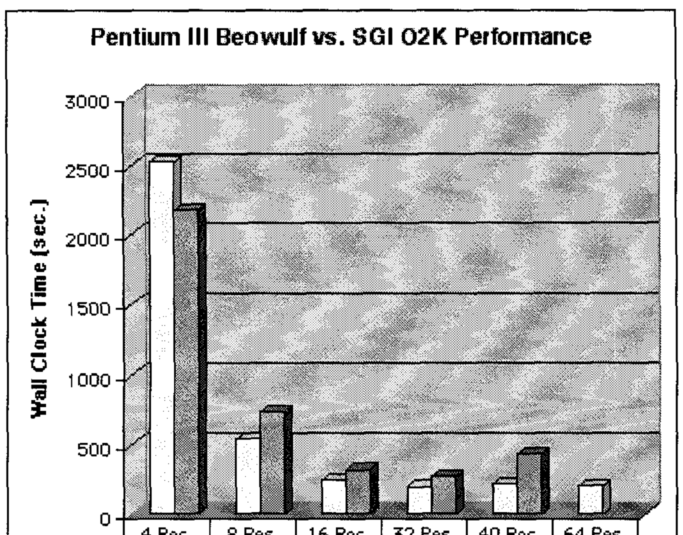
- Maximize exchanges at each level without repeated calculations
- Reduce data volume at each level with full-duplex communication
- Much fewer broadcasts are required to support an arbitrary number of processors
- Processors which do not contribute to the calculation at a given level are idle



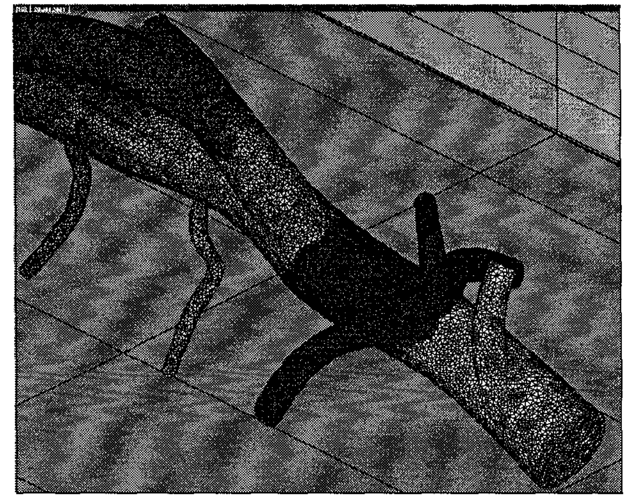
Maximum number of pairwise exchanges are performed

Performance

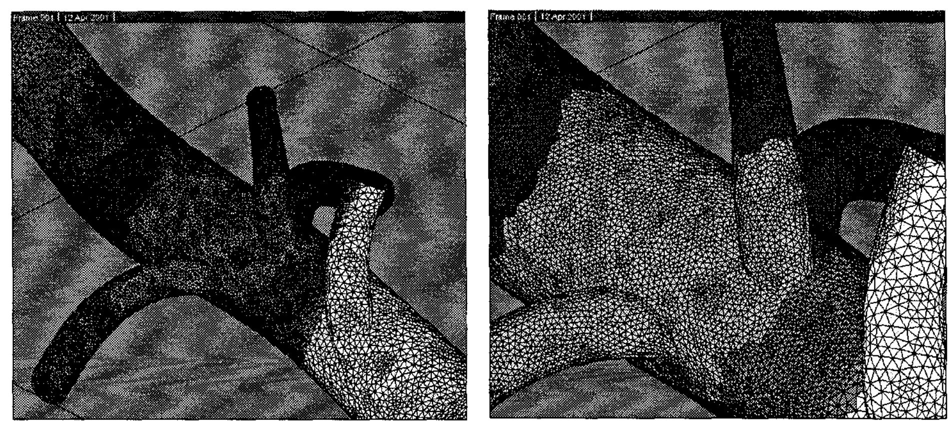
- **Pentium III Beowulf Cluster vs. SGI O2K Migration**
  - Beowulf performs well, but network dominates with increasing processors
  - O2K is also affected for large (>30MB) messages



	4 Pes	8 Pes	16 Pes	32 Pes	40 Pes	64 Pes
SGI O2K	2527.87	537.02	244.85	195.89	215.88	206.91
Beowulf	2176.99	731.46	321.21	275.62	435.12	



Artery Segment with 1.1 Million Elements



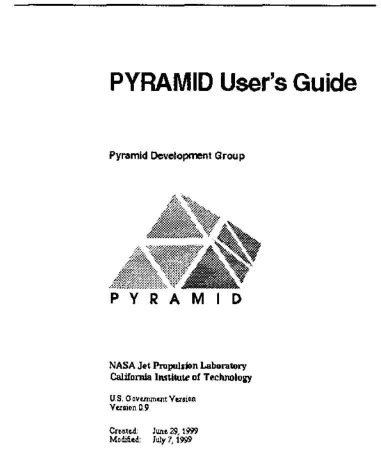
Adaptively Refined Artery Segment with 2 Million Elements

Note : New migration algorithms are applied

Next Generation Features

- **Development is User-Driven**
  - Used for adaptive refinement of multi-scale meshes for active device modeling
- **Additional Work Directions**
  - User-controllable boundary zone definition
  - Interpolation methods among mesh levels
  - Straightforward approaches for incorporating error estimation
  - Coarsening
- **Demonstration Release**
  - hpc.jpl.nasa.gov/APPS/AMR

Note : Functionality is limited in demo release



PYRAMID User's Guide

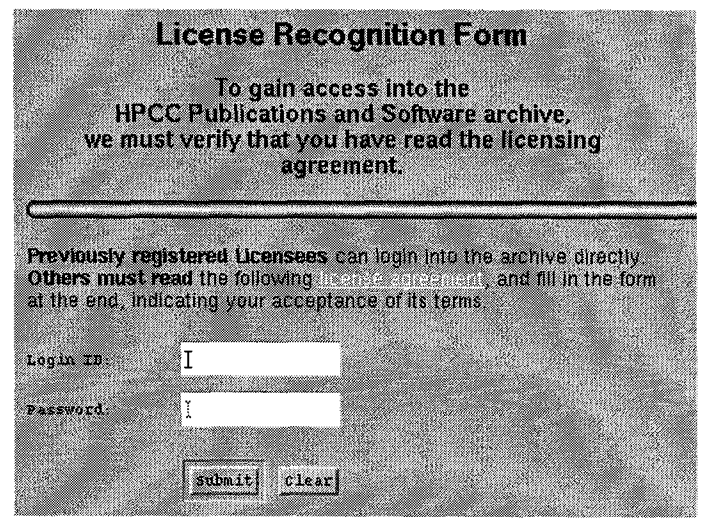
Pyramid Development Group

PYRAMID

NASA Jet Propulsion Laboratory
California Institute of Technology

U.S. Government Version
Version 0.9

Created: June 29, 1999
Modified: July 7, 1999



License Recognition Form

To gain access into the HPCC Publications and Software archive, we must verify that you have read the licensing agreement.

Previously registered Licensees can login into the archive directly. **Others must read** the following license agreement, and fill in the form at the end, indicating your acceptance of its terms.

Login ID:

Password:

Submit Clear